Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Auto-response Email based on User Habits with Privacy Model Approach and Cloud Adoption

Gabriella Vindy Kawuri[1] ✉ Ridi Ferdiana[2] and Lukito Edi Nugroho[3]
[123]Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia
Corresponding Author: Gabriella Vindy Kawuri, **E-mail**: gabriellavindy@mail.ugm.ac.id

| ABSTRACT
Email is an application that is used as a means of sending letters via a computer network or the internet. Many people whose work is done relate to other people, and email is the main channel through which work and related information can be distributed. An email has a feature that is to enable automatic responses. This feature is in the form of an automatic reply that will be sent in response to emails received when the email user is unable to respond. However, this feature still uses the default email to reply to receive emails. So in this study, the researcher proposes a new system to develop this feature by using user habits when the user replies to incoming emails. Using text processing techniques and sentiment analysis, we build a retrieval algorithm that finds similar cases outside of exact matches. This study uses a *Cloud Adoption* and is designed to use *Power Automate* to improve the performance of the system.

| KEYWORDS

Email, auto-response, cloud adoption, power automate

| ARTICLE INFORMATION

ACCEPTED: 17 November 2022 **PUBLISHED:** 01 December 2022 **DOI:** 10.32996/jcsts.2022.4.2.15

1. Introduction

Electronic Mail (email) is an application that is used as a means of sending letters via a computer network (internet). By using e-mail, data transmission activities such as correspondence are easier, faster, and cheaper than using human services (postal). As a result, the number of emails sent and received continues to increase.

This interaction via email is usually also carried out in the development of a product or software development. In this development, there are methods that are usually used, such as *Agile Development Methods*. This agile method will facilitate interaction between the *Development Team* and clients regarding needs that can change at any time. One well-known model of agile methods is *Scrum*.

Scrum is a framework that is carried out iteratively until the managed product or application developed has met the appropriate needs. Each iteration will involve the team in the development cycle, such as planning, needs analysis, design, implementation, testing, and documentation to *maintenance* so as to be able to adapt to any changes that can occur at any time.

Implementation of *Scrum* can be done *face to face* but can also be done via email, which is easy to reach. *The Scrum Team* is expected to be able to deliver products on a regular basis, maximize coordination, prioritize work and receive input or send difficulties via email. In addition, there is a *Daily Scrum* which is conducted every day. Because this activity is carried out in a short time, it requires a quick response, including when the team does it via email.

The email system's well-known feature to support the Daily Scrum for Scrum Team is enabling the automatic response feature. The auto-response feature provides an e-mail server to automatically send e-mail replies in response to e-mails received on behalf

of the user. This is important for the *Scrum Team* because it can make it easier for those who have a lot of emails that come in every day, where most of these emails are related to some common problems. As for *Daily Scrum* itself, it is only done briefly and has a time limit for each day. So that when the user is unable to check their email, a short auto-reply email is sent in response to the received email then the sender of the received email is notified of the user's inability to respond. With this, it will not hinder the *Scrum Team* towards the *Sprint Goal*.

The automatic response feature found in existing applications still uses the default message to reply to received emails. So that the message data used is the same for each response from users who use the application. Meanwhile, each user, especially the *Scrum Team*, has different responses, which cannot always be set in the automatic response setting feature. Specifically, the different responses are meant to use the previous reply from the user to respond to future incoming emails. This can be obtained because most of the time, the incoming email has something in common, so there will be a similar reply that can be used. So that the recipient of the email will feel more if the email was made by the sender as usual.

Therefore this research wants to develop this feature for the better that can be used by email users, including the *Scrum Team,* by taking the benefit of reusing its own replies in the past if the messages are similar. This can improve the experience of replying to emails, as users don't have to start with a default message or a blank message. It can also reduce the time and energy it takes to compose messages from scratch or browse through emails to find similar messages.

In this study, it is formulated how to develop cognitive applications to provide automatic response features based on user habits; So that there is an automatic response when the user gets an incoming email when the user is not opening his email, as well as how to maintain the security of messages from private users so that they are not sent automatically. In addition, in development, so that it is always directed to the problems at hand, it has a problem limitation; namely, this automatic response will only be used on email.

The results of this study are expected to be useful when an email user is not actively using email but there is an incoming email; it will be replied immediately according to the user's habits. That way, the response for each email does not depend on the default provided by the system. This process will reuse replies from existing emails that are similar to those that just came in. In particular, this will make it easier for users, especially the *Scrum Team*, to do *Daily Scrum*. Because working together as a team every day requires communication with a fast response, in this case, via email. So this will help the team progress towards the *Sprint Goal* and produce plans for the *Scrum Team*.

2. Materials and Methods

This research produces a cognitive application for automatic responses to emails based on user habits in using email. In order to support this writing, the author uses a research method, namely a literature study, to find references related to the writing topic discussed from information sourced from theoretical books, journals, and other scientific articles in order to increase knowledge, insight, and develop theoretical concepts.

A. **Email Autoresponse**

Email autoresponse is a system that can automatically respond to incoming emails to an email address (*mail server*) and then perform *autoreply*. *Autoresponders* have been widely applied in today's internet networks, such as those done by Yahoo and Gmail. When we register as a user in (Gmail), we immediately get an email containing a welcome from Gmail mail itself. This is one application of mail *autoresponse*.

In sending incoming emails, a *mail processing program* whose function is to filter emails to run a script is used, one of which is *Procmail*. *Procmail* is a *mail processing program* on UNIX; it can help complete the following tasks [McCarthy, 2001];

1) Filtering incoming emails (e.g., separating mailing-list messages from other messages) is very useful for *preprocessing* large numbers of incoming emails.
2) Sort incoming emails by criteria, such as sender, message size, or *keywords* in the *subject field*.
3) mechanisms *autoreply,* such as replying to emails to those who sent messages automatically.

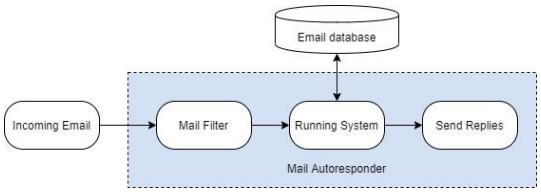

FIGURE I. Mail Autoresponse System

Another method of personalizing auto-reply messages requires a pre-formatted message to be received from the sender, a pre-formatted message containing the sender's name. The sender's name is accurately parsed from the received pre-formatted message and stored in *a database*. After receiving further messages from the sender, a database search is performed to retrieve the previously stored sender name, and auto-reply messages can be personalized using the retrieved username. One of the problems with this approach is that it is impractical to require the sender to send a message that has been formatted to potential recipients to have an accurately personalized auto-reply message [Mark, 2005].

B. Scrum Methodology

Development method *Scrum* was proposed in 1995 by Schwaber [1997], it became clear to most professionals that software development is not something that can be planned, estimated, and completed successfully using *'heavy'*. The main purpose of *Scrum* is to inspect and adapt, which means to see the existing problems and adapt to those problems. Software development *Scrum* emphasizes taking every step of software development in a nutshell [Deemer, 2012].

According to Cervone [2011], *Scrum* is formed by three main elements: roles, processes, and artifacts. Among the roles, there are *Scrum Master*, *Development Team,* and *Product Owner*. Among these processes, there are *kickoffs*, *sprint planning*, *sprints*, *daily scrums*, and *retrospective sprints*. Finally, artifacts include *product backlogs*, *sprint backlogs*, and *burndown charts*. Although not cited by Cervone, among these processes, there is also a retrospective meeting [Schwaber, 2017].

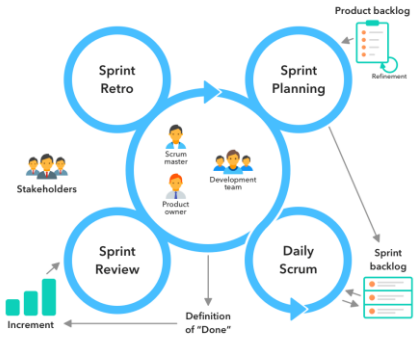

FIGURE II. The Scrum Process

At the heart of *Scrum* is the idea that many processes during development are unpredictable. Hence, it handles software development in a flexible way. The only two parts that are fully defined during a software development project are the first and final phases (planning and closing). Among other things, the final product is developed by several teams in a series of *blackboxes* flexible *'sprints'*. This ensures that the final product is being developed with a high probability of success, even in a constantly changing environment. This environment, which includes factors such as competition, timing, and financial pressures, maintains its influence on development until the closing phase.

In *Scrum*, each *sprint* begins with a daily planning meeting. During this meeting, the *Product Owner* and members of the development team agree on what is to be accomplished. To determine what can be achieved during each *sprint*, the development team has greater consideration, while the *Product Owner* is responsible for determining the criteria that will determine whether the work can be approved and accepted [Sletholt, 2011].

Sprints in *Scrum* are essential for providing specifications to complete the work being developed by the team over a fixed duration or timeframe. Possible adjustments to scope and stakeholders can be considered during the planning process and sometimes during the *sprint* itself if major issues develop that impact delivery or completion [Torrecilla-Salinas, 2016]. Duration *Sprint* is determined by *Scrum Master*, who serves as a facilitator. There are several advantages and disadvantages of *sprints* shorter/longer; usually, *sprints* last for 1 to 2 weeks, and *sprints* last for 3 to 4 weeks. Regardless of the duration of the *sprint*, *Development Team*

must complete the *sprint* with the proposed *Product Owner,* who determines whether he or she accepts or rejects the resulting work.

Team members are required to participate in *Daily Stand-Up Meetings,* so each member is an important part of the group. Usually, the members can discuss their difficulties, progress, success, or obstacles to completing *the sprint* [Sutherland, 2015]. *Daily Stand-Up Meetings* focus on '*blockers*' to get the job done.

A *sprint review* is held at the end of each *sprint* as an opportunity for scheduled inspections and adaptations of what has been developed. The team must produce a product, feature, or improvement, such as high-quality software that can be developed for *Human-Computer Interaction*activities, database searches, e-commerce activities, and several others at the end of the *sprint*. After that, a retrospective meeting was held where the *Scrum Master*, team, and *Product Owner* discussed the main events in *the sprint* and what actions would be taken to improve them in *sprint* [Rubin, 2012].

C. Cloud Adoption

Cloud Security Alliance (CSA) provides a definition of *cloud computing* as a model for enabling network access from anywhere, conveniently, on demand to the set of computing resources to be configured, such as networks, *servers*, storage, applications, and other services [Brunette, 2009].

Provisioning models *cloud* to represent any particular combination of IT resources packaged together collectively referred to as the SPI (*Software, Platform, Infrastructure*) model [Erl, 2013].

1) *Infrastructure as a Service (IaaS):* The capabilities provided to consumers are providing processing, storage, networking, and other basic computing resources where consumers can deploy and run operating systems and applications as needed.
2) *Platform as a Service (Paas)*: The capability provided to the consumer is to deploy to the *cloud* consumer-built or acquired applications using programming languages, services, and tools supported by the *cloud*.
3) *Software as a Service (SaaS):* The capability provided to consumers is to use the provider's applications running on a *cloud*.

Cloud-based infrastructure is fundamentally changing the way organizations find, use, and secure technology resources. While the cloud offers tremendous flexibility in design choices, organizations need a proven and consistent methodology for adopting cloud technologies to ensure success. *Cloud Adoption* is a means of ending a successful start well before a cloud platform vendor is chosen. *Cloud Adoption Framework* helps meet these needs by guiding decisions along the cloud adoption journey to achieve desired business outcomes.

Cloud Adoption Framework is a lifecycle framework that enables cloud architects, IT professionals, and business decision-makers to achieve business goals. It provides documentation, technical guidance, best practices, and tools that help align business, organizational readiness, and technology strategies for the cloud. The following diagram outlines how the framework uses the methodology as an approach to help customers take their cloud adoption journey:

FIGURE III. Cloud Adoption Framework

D. Privacy Model

The privacy policy to be enforced by this model can be described informally as follows: Users can only have access to personal data if this access is necessary to perform their current duties if the user is authorized to perform these tasks. The user can only access the data in a controlled manner by performing a (certified) transformation procedure for which the current user task is authorized. In addition, the purpose of his current assignment must match the purpose for which the personal data was obtained, or there must be consent by the data subject. This formal task-based privacy model is defined as *the state machine model*. The privacy model contains *state variables, invariants, constraints* (privacy properties), and *state transition functions* (model rules).

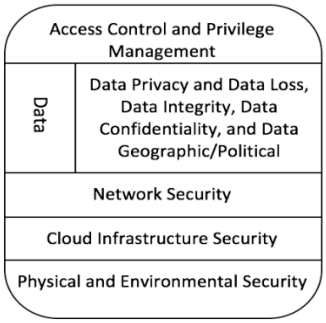

FIGURE IV. Cloud Security and Privacy Model The

The cloud security and privacy model (CSPM) is a layered model, which is divided into five layers: *Physical and Environmental Security, Cloud Infrastructure Security, Network Security*, Data, and *Access Control and Privilege Management,* as shown in Figure IV [El Makkaoui, 2016].

E. Power Automate

Power Automate formerly known as *Microsoft Flow,* is a *Software as a Service* (SaaS) offering derived from *Azure Logic Apps*, a tool known as IT developer tools, and they share the same codebase. Both make use of *Workflow Definition Language* to build expressions, and there are near-perfect feature similarities between the two tools under discussion with few exceptions. Feature parity clearly means that they share the same functionality and feature capabilities. In fact, streams can be easily exported from *Power Automate* and deployed as *Azure Logic Apps* [Pearson, 2020].

A development platform *low-code* can be used to build a *workflow*. These development tools are supported by a simple and intuitive user interface. *The interface* contained in the *tool* is equipped with more than 275 *connectors* and thousands *of pre-built*. *Connectors, actions,* and templates allow "*Citizen Developers*" to easily connect to applications created and some of the popular services during this development process [Pearson, 2020].

With *Power Automate*, you can automate time-consuming manual tasks with built-in AI capabilities, giving you more time to focus on strategic, high-value opportunities.

1) Increase productivity and accuracy
2) Free time to focus on strategic opportunities
3) Achieve more with less effort
4) Streamline business processes
5) Add intelligence to automated processes
6) Predict results to improve performance

F. System Analysis

The next computational steps required by utilizing *Power Automate* in developing automated responses to emails are described below:

1) *Incoming Email Identification*

Before starting to develop the flow for auto-response, a series of text-prepossessing steps on the original email dataset were executed. For processed emails, we store only matched email pairs (*received email–reply email*) so that we obtain the relationship between the received email and the email that was replied to and store it in the database. Then when a new email is received, the first step is to identify the sender of the email. If there is an email in the form of spam, then the process will not continue. If it is not a spam email, the system will retrieve data from the email.

2) *Email Classification*

The domain faced in this study, namely email communication, has a collection of text-based messages. Although email messages have a clear structure made up of sender, recipient, date, subject, and body, the biggest sources of information lie in the subject and body. Moreover, these fields can form semi-structured or even unstructured texts, making it a challenge to extract useful knowledge from them.

We characterize several factors that affect email replies. For replies, we calculate the reply rate, which is the percentage of emails that receive a reply.

Because the system will take advantage of previous email replies to answer incoming emails, it is important to perform several techniques to analyze message content to remove distractions and identify meaningful contexts. Then the email will be classified into various categories based on several keywords. This will make it easier for future messages by leveraging those categories to find previous similar messages and then reusing the same replies with that user's habits. So it is necessary to do an analysis to normalize the text.

Data is collected, which includes a combination of textual and numeric data that has been retrieved from the email. Only textual data is considered separate from other data to prepare for the data cleaning step. After separating the textual data, we have to clean it up. Cleaning is the process of standardizing and removing irrelevant characters and text. This step consists of the total process of preparing the text for the modeling operation.

The result of cleaning email messages that have been done will be a *bag of words*. *A bag of words* is a simple display used in natural language processing and *data retrieval*. Also known as the vector space model. In this model, multi-word sets where text such as sentences or documents are displayed as packages, regardless of grammar or even word order, and maintain multiplicity. This model is a safe method for understanding the text, which is used to retrieve information for the text corpus by ignoring the sentence structure.

A keyword search model *bag of words* is applied to count the number of times each word appears in an email message. These keywords are used to describe email communication patterns. This means that one can use the keyword index to determine the category of email messages. The system has determined categories based on the email dataset used to compare with new emails.

3) *Email Sentiment Analysis*
Sentiment analysis aims to automatically extract and predict sentiments or emotions conveyed through text documents, which is an applied field of study in the field of *text mining*. According to the natural characteristics of the text data set, sentiment analysis tasks are mainly categorized into document, sentence, and aspect level analysis. This platform is used to achieve positive, negative, and neutral attitudes.

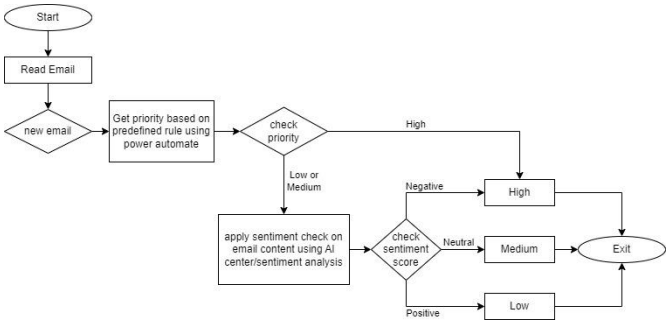

FIGURE V. Cloud Adoption Framework

The sentiment analysis process in *autoresponse* is carried out using data from the *bag of words* that already exist in the previous process. The data is used to implement topic modeling using the best algorithm. But before implementing the model, we need to define the input parameter values. To achieve this, the model is evaluated with different parameters to select the most appropriate parameters to ensure optimal model output. An index will be needed to confirm the results of the modeling algorithm by using a coherence score. Coherence score, as an evaluation criterion, is a topic identification by calculating the level of semantic similarity between words that have a high score on a topic. It also helps to measure differences between topics which can be interpreted semantically in statistical inference.

Next, there is polarity calculation and sentiment analysis; in this section, it is necessary to determine the average polarity of received email messages. This message can be positive or negative or neither and can be classified as a neutral opinion. The input of this step is the dictionary of words selected in the previous step, as well as special pre-processing suitable for this operation in the data cleaning step. The calculations obtained from this section go into the sentiment analysis process. It is a study that tries to express the feelings, behaviors, opinions, and analyzes of different people regarding entities and their characteristics. The output of the operation of the previous step is calculated for each message received and prepared in tabular form by showing a better comparison between different messages.

4) *Replying to Emails*

The final step in the system is to send a reply to the new incoming email. Because the system is an *autoresponse* that will immediately send a response when an email comes in, this research will ignore the user's habit of responding to emails. So it only focuses on user habits in email reply actions. Countermeasures are to determine the possibility of whether the incoming message will receive a reply or not by looking at the existing data. Or in other words, the incoming email has the same category as the previous email belonging to the user that has been done in the previous process. That way, the system will be able to retrieve response data that matches the incoming email. Then the system will immediately send the response based on the content of the message that the user usually uses.

3. Results and Discussion

The results of the research are presented in this chapter, divided into three phases of design research. First, the results from the problem identification stage are presented, then the results from the design stage, and finally, the results from the evaluation stage.

A. Problem Identification Phase

The purpose of the problem identification phase is to understand the problems encountered when utilizing *Power Automate* cloud-based *autoresponse* that will be implemented in email. This section will discuss the results of the modeling and evaluation of the process.

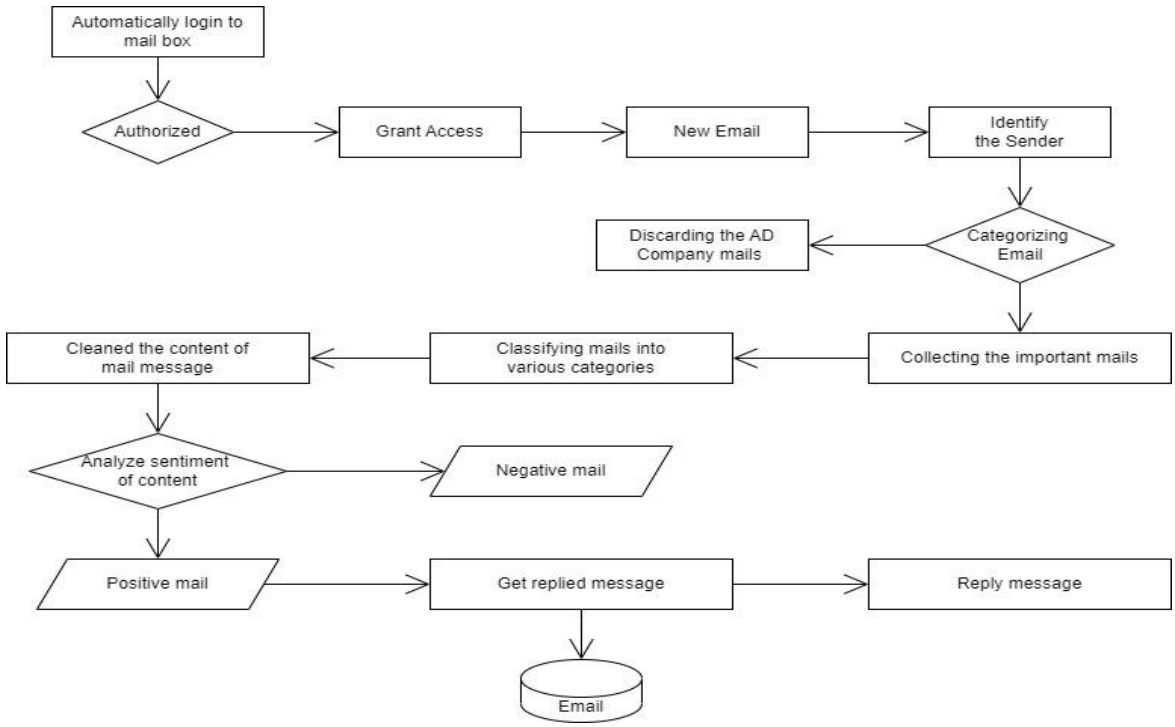

FIGURE VI. Flowchart of the Email Autoresponse System

Figure VI describes the *flowchart* of the *autoresponse* proposed email. It also shows the order in which activities are executed on the flow.

The app starts working when the user initiates a reply action for a new incoming message. It then tries to retrieve similar previous incoming emails (known as cases), reuses the responses, and then uses them as the basis for replies to those emails. Furthermore, the system also analyzes the sentiment of the message to maintain the security of the email. The process is complete when the response has been obtained and sent. Replied emails and replies are considered as resolved issues and kept on a case basis for the future.

Before the process is applied to the system in *Power Automate*, the first step is to determine what features will be built according to their priorities. The features to be built are shown in Table I. Where there are 4 features to be developed on the system, the list of features is then referred to as the *Product Backlog*.

TABLE I. Product Backlog List

No.	Feature Description	Priority
1	Identify the sender of the email	Medium
2	Classify the email into several categories	High
3	Sentiment analysis of the email content	High
4	Reply to email	High

B. Design Stage

At this stage, the *sprint* table *product backlog sprints* resulting *sprints* with consideration of the *backlog*rules *scrum*, which will later become *the sprint backlog*. The following are the stages of the event (*scrum event*) in each *sprint*.

Sprint 1: Identify the email sender

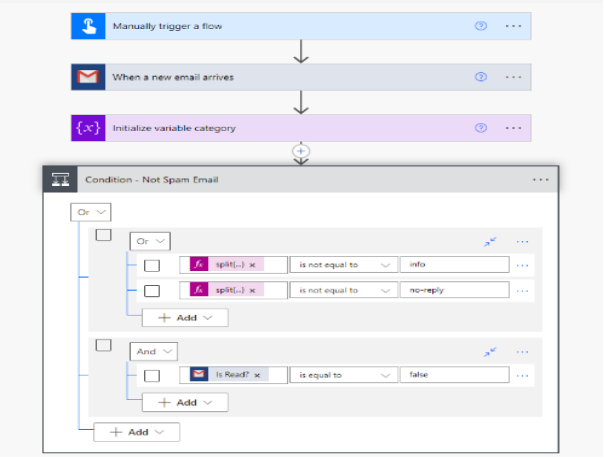

FIGURE VII. Email Sender Identification Flow

Figure VII. provides an overview of the flow when email arrives. When there is a new email that comes in, the system will capture data from the email, such as email address, sender name, recipient, date, subject, and also message content. To identify whether the email is spam or not, it will look at the email address.

Sprint 2: Classify emails into categories

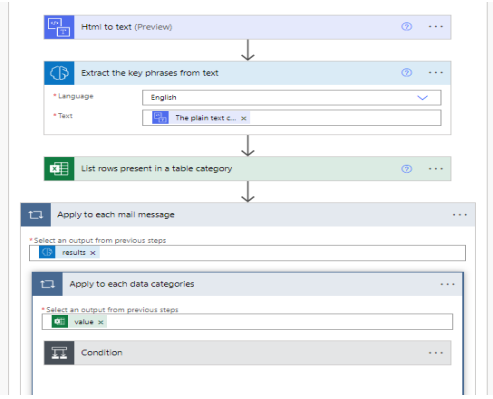

FIGURE VIII. Message Classification Flow

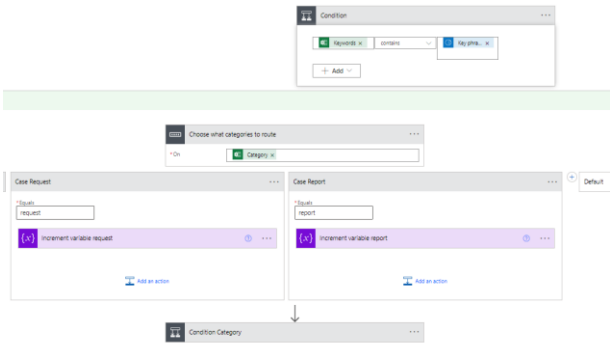

FIGURE IX. Flow Defines Message Category

In this application, we handle text-based email messages. Therefore, it is important to employ several techniques to analyze message content, including removing distractions and identifying meaningful contexts. First, we will convert the message into text form because the incoming message is still in HTML form.

Next, we performed a lexical analysis to normalize the text, including the treatment of punctuation, numbers, and hyphens, as well as upper and lower case letters. This process is followed by converting the stream of text in the document, which is usually separated by spaces, into a stream of candidate words to be indexed. Once a stream of potential words for the use of index terms has been generated through lexical analysis, some may prove to be less influential in the context of the document. This happens when they appear too often in a document and can become useless in retrieval tasks. So, we removed the words commonly known as *stopwords*.

In addition, we derive the word from its root. This method is generally applied to reduce word variability; thus, more matches can be achieved. For example, strings such as *buy*, *buys*, and *buying* can be reduced to their root, *buy*. This can reduce the number of index terms so that less number of distinct words (*word roots*) are stored. The collection of indexes obtained becomes *keywords* which will then be compared with *keywords* from each pre-determined category.

Sprint 3: Sentiment analysis of email content

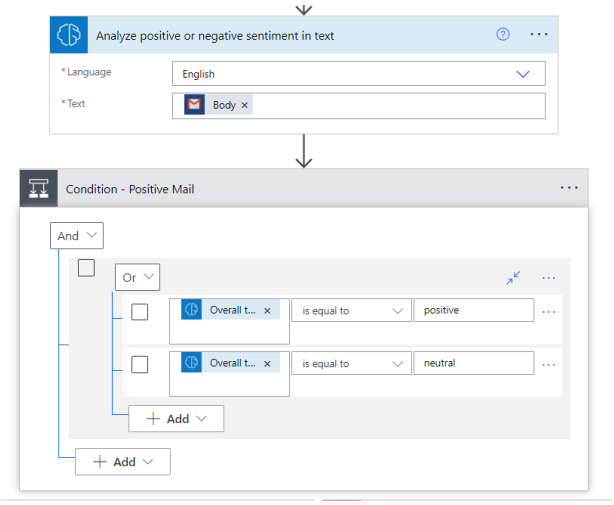

FIGURE X. Email Sentiment Analysis Flow

At this stage, the system will perform a sentiment analysis of the content of the new email. From this process, the system will calculate the average polarity of the messages used to determine which emails are positive, neutral, or negative.

Sprint 4: Reply to email

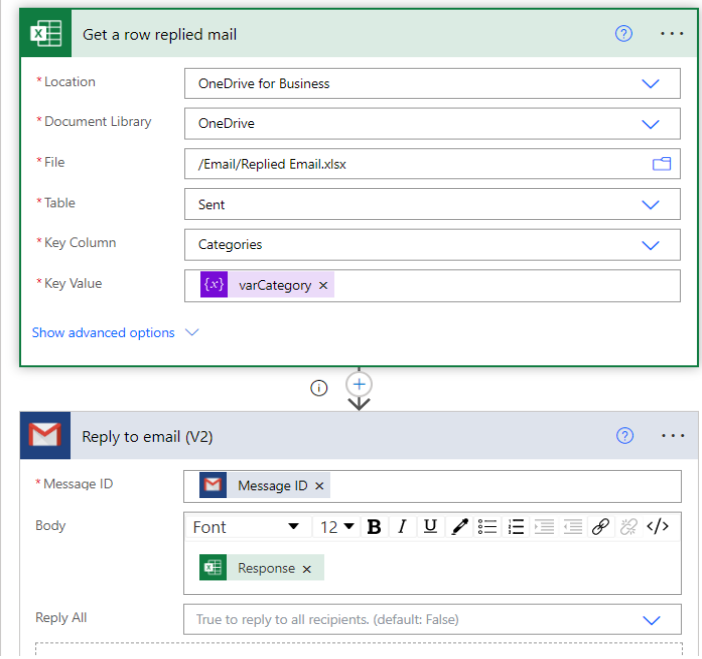

FIGURE XI. Flow of Replying to Emails

At the end of the process, if the results of the sentiment analysis on the email message are positive or neutral, the system will retrieve data according to the categories that have been determined in the previous process. Then the system will send a reply message using the data obtained, such as the flow shown in Figure XI.

C. Evaluation Phase

Data for this study came from the Enron email dataset that had more than 200 email conversations. The data for each email is stored in the system database. Furthermore, system testing is carried out using a similarity measure approach embedded in the 200 email datasets. To measure the quality and goodness of *autoresponses*, the gold standard is used as a reference.

FIGURE XII. Keyword Extraction Results from Email

In the framework experiment, *Power Automate* extracted new emails and got the keywords, as shown in Figure XII. Then the selected keywords are checked with the keyword selection that has been collected and used to determine the category of the email.

FIGURE XIII. Sentiment Analysis Output

Figure XIII shows the output of sentiment analysis run by *Power Automate*. These results show that the polarity value for incoming email is a positive probability value of 0.83, a negative probability value of 0.02, and a neutral probability value of 0.15. That way, the email is positive so the system can proceed to the next process, which is to send a reply, as shown in Figure XIV below.

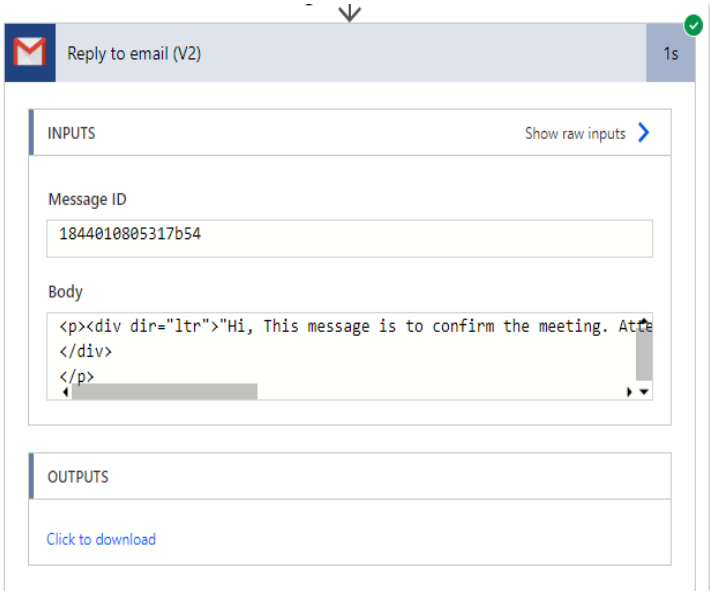

FIGURE XIV. Sending Email Reply

4. Conclusion

In this study, we have developed a cognitive application to provide an automatic response feature based on user habits. Using text processing and semantic analysis techniques, we build a retrieval algorithm that finds similar cases beyond exact matches. This approach can be used to ensure the security and privacy of the user's email. We have presented a systematic guide to identification and classification using *Power Automate*. The algorithm takes similar previous messages and replies according to the incoming messages based on important keywords. The evaluation process ends with the automatic response to the received email and the suitability of the reply data with user habits.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Brunette, G, and Mogull, R. (2009). Security Guidance Critical Areas of Focus for Critical Areas of Focus in Cloud Computing V2.1, *Cloud Security Alliance*.

[2] Cervone, HF. (2011). Understanding Agile Project Management Methods Using Scrum," *OCLC Systems and Services, 27*(1). 18-22, 2011.

[3] Deemer, P, Benefield, G, Larman, C and Vodde B. (2012). The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum," Info Queue

[4] Erl, T, Mahmood Z and Puttini, R. (2013). Cloud Computing Concepts, Technology and Architecture, Prentice Hall.

[5] El Makkaoui, K, Ezzati, A, Hssane AB and Motamed, C. (2016). Data Confidentiality in The Word of Cloud, *Journal of Theoretical and Applied Information Technology*. 84. 3, 2016.

[6] McCarthy, M. (2001). The Procmail Companion, London: Pearson Business.

[7] Mark, D John B and Fernando, P. (2005). Reply Expectation Prediction for Email Management', In the 2nd Conference on Email and Anti-Spam, Stanford University: California, USA, 2005.

[8] Pearson, M, Knight, B, Knight, D and Quintana, M. (2020). Pro Microsoft Power Platform: Solution Building for the Citizen Developer, Florida: Apress,

[9] Rubin, KS. (2012). Essential Scrum, in *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, Michigan, Pearson Education

[10] Schwaber K and Suttherland, J. (2017). The Scrum Guide: The Definitive The Rules of the Game," *Scrum.Org and ScrumInc*

[11] Sletholt, MT, Hannay, J, Pfahl, D, Benestad HC and Langtangen, HP. (2011). A Literature Review of Agile Practices and Their Effects in Scientific Software Development, in *Proceedings - International Conference on Software Engineering*, 2011.

[12] Sutherland, J. (2015). SCRUM: The Art of Doing Twice the Work in Half the Time, New York: Crown Business.

[13] Torrecilla-Salinas, CJ, Sedeño, J, Escalona MJ and Mejías, M. (2016). Agile, Web Engineering and Capability Maturity Model Integration: A systematic literature review," *Information and Software Technology, 71*(92)–107, 2016.